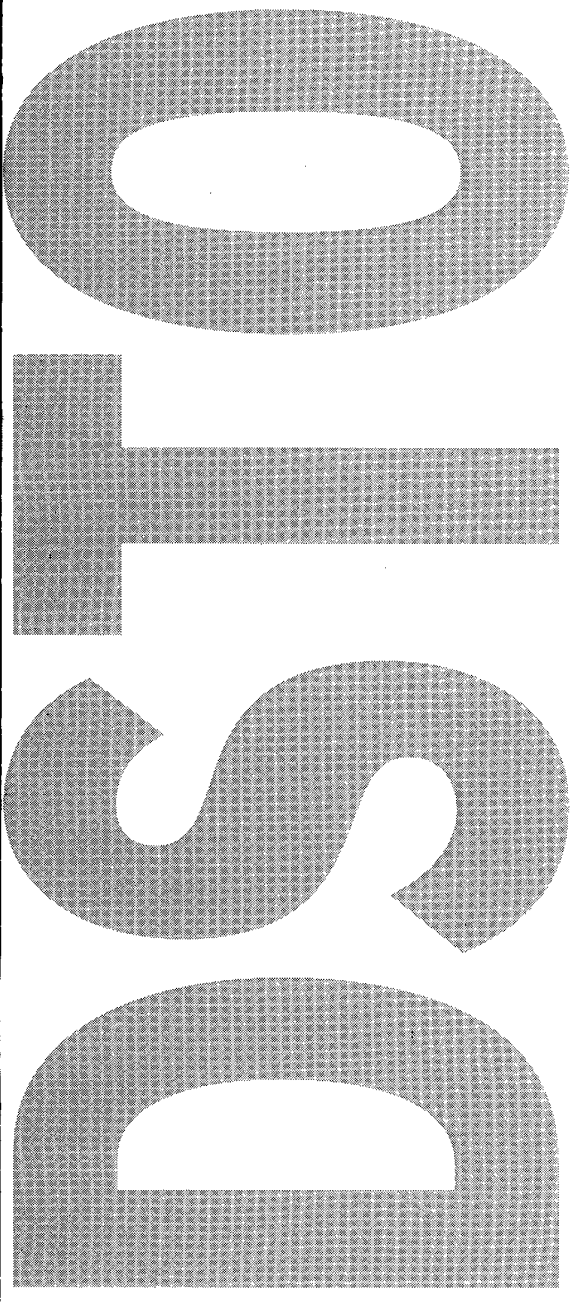
Oct 2002



Interfacing COTS Speech Recognition and Synthesis Software to a Lotus Notes Military Command and Control Database

Oliver Carr

DSTO-TR-1358

221 140

Interfacing COTS Speech Recognition and Synthesis Software to a Lotus Notes Military Command and Control Database

Oliver Carr

Command and Control Division
Information Sciences Laboratory

DSTO-TR-1358

ABSTRACT

Speech recognition and synthesis technologies have become commercially viable over recent years. Two current market leading products in speech recognition technology are Dragon NaturallySpeaking and IBM ViaVoice. This report describes the development of speech user interfaces incorporating these products with Lotus Notes and Java applications. These interfaces enable data entry using speech recognition and allow warnings and instructions to be issued via speech synthesis. The development of a military vocabulary to improve user interaction is discussed. The report also describes an evaluation in terms of speed of the various speech user interfaces developed using Dragon NaturallySpeaking and IBM ViaVoice with a Lotus Notes Command and Control Support System Log database.

Published by

DSTO Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR- 012-484
October 2002

APPROVED FOR PUBLIC RELEASE

Interfacing COTS Speech Recognition and Synthesis Software to a Lotus Notes Military Command and Control Database

Executive Summary

There are 1300 applications using Lotus Notes in the Australian Defence Organisation. IBM is convinced that Lotus' COTS software with appropriate military amendments can play a significant role in military crisis applications. The development of techniques and tools to integrate Lotus Notes databases with COTS speech recognition and synthesis aims to increase user efficiency and reduce the workload of administrative tasks. A database containing military acronyms and abbreviations is required to adapt any COTS speech recognition and synthesis product for use in a military environment. A military vocabulary for both Dragon NaturallySpeaking and IBM ViaVoice was created using a Defence Intranet document defining about 4,000 acronyms and abbreviations. Future work is in progress to develop a Lotus Notes database that categorises 40,000 acronyms and abbreviations provided by Defence Materiel Organisation.

A user interface that incorporates speech recognition and/or speech synthesis is called a speech user interface. The development of speech user interfaces using Dragon NaturallySpeaking and IBM ViaVoice are described for a Lotus Notes Command Support System Log database. The speech user interfaces developed for Lotus Notes use a technique called Scripting Commands. These commands are defined for a window of a particular application. The resultant actions of these commands mimic keystrokes that are sent to the application. The speech user interfaces developed for a Lotus Notes Command Support System Log database using both Dragon NaturallySpeaking and IBM ViaVoice were evaluated. Dragon NaturallySpeaking was found to respond to commands much faster than IBM ViaVoice. However, IBM ViaVoice has been found to have slightly higher recognition accuracy than Dragon NaturallySpeaking. Dragon NaturallySpeaking also has better development tools available for Scripting Commands than IBM ViaVoice.

The US Army Communications-Electronics Command has a strategy of Adopt, Adapt and Develop for integrating COTS products into their systems. They advocate that programming standards should be used in software systems as much as possible. The Java Speech API provides a standard interface to speech recognition and synthesis technology. This standard has been developed by Sun Microsystems in cooperation with other companies. IBM's implementation of the Java Speech API, Speech for Java, was used to interface IBM ViaVoice with a Java version of the Lotus Notes CSS Log database. There is no implementation of the Java Speech API for Dragon

NaturallySpeaking. A software package called J-Integra was used to create an interface between Java and components of the Dragon NaturallySpeaking SDK.

The design of a speech user interface for a Windows based application could use either Dragon NaturallySpeaking or IBM ViaVoice depending on the capabilities of the two products, cost of licenses, development time and maintenance requirements. The use of high-level languages describing the user level transaction design could in the future automatically integrate an application with any COTS speech recognition and synthesis product. The integration of speech recognition, speech synthesis and artificial intelligence technology could conceivably result in a computer that understands speech and communicates as humans do.

Author

Oliver Carr
Command and Control Division

Oliver has worked in the Human Systems Integration group since 1998. Previously he completed a BSc.(Ma .& Comp.Sc.) at the University of Adelaide and Honours in Pure Mathematics part-time from 1998 to 2000 whilst at DSTO. His current research interests include speech user interface languages and techniques, developing tools for managing military vocabularies for speech recognition products, using speech recognition with collaborative meeting support tools and developing advanced query tools using information extraction techniques from free text fields in Lotus Notes databases.

Contents

1. Introduction

The continual improvements of PC technology in recent years have made speech recognition and synthesis commercially viable for individuals and companies. In coming years the microphone will become essential equipment for a PC (Behr, 2002). How the user interfaces built with this technology and how it is used is uncertain (Rosenfeld et. al., 2001).

Speech user interfaces were developed with a Lotus[1] Notes database using two leading COTS speech recognition and synthesis products. Lotus Notes is widely used within the Australian Defence Force for messaging and communication. One of the research and development objectives of the Deployable Joint Force Headquarters (DJFHQ) is to investigate speech user interfaces to a Lotus Notes Command Support System (CSS) Log database. This database is used within the Joint Personnel Manpower and Logistics branch within the DJFHQ for logging events, requests and orders. We aimed to improve the front-end user interface to the Lotus Notes CSS Log database and to investigate the additional use of COTS speech recognition and synthesis software. At the Crocodile 99 Exercise, operators of the Lotus Notes CSS Log database were observed. The problems the operators had with the user interface were noted. To address these problems a Java CSS Log user interface to the Lotus Notes CSS Log database was designed and developed. The various programming techniques provided by two market leading COTS speech recognition and synthesis products, Dragon NaturallySpeaking and IBM ViaVoice, were evaluated for speech input and output in both user interfaces.

IBM ViaVoice and Dragon NaturallySpeaking are large vocabulary continuous speech recognition products available in several languages. These products have user vocabularies in the order of 200,000 words. The user can also add words and improve the internal language models by analysing their previously written Word and Excel documents. The speech synthesis component of IBM ViaVoice was used to issue auditory warnings and instructions in the Lotus Notes CSS Log. The development of a military vocabulary for both IBM ViaVoice and Dragon NaturallySpeaking to aid user interaction is also discussed.

The Lotus Notes CSS Log database contains several forms for data entry. These forms contain fixed fields with finite sets of possible values and free text fields. We aimed to allow the user to fill out all these fields in a form by voice under the following guidelines. At any time the user can do any of the following:

- For fixed fields, say the field name followed by an item of the field, which selects the item in the field.
- For fixed fields, say the field name which opens a dialog to choose the item.
- For free text fields, say the field name, which places the cursor in the field.
- For fixed fields, say the field value, which selects the item in the appropriate field.

[1] The following trademarks appear in this report and are property of their respective owners: Java, IBM, ViaVoice, IBM Viavoice, Dragon Systems, Dragon NaturallySpeaking, NatLink, PutWords, ScanSoft, Lotus Software, Lotus Sametime, Lotus Notes, J-Integra, Sun Microsystems, Word, Excel, Microsoft, Swing, ActiveX and Common Object Model (COM).

- Say basic Windows management commands such as *"close window"* or *"file close"*.
- Dictate into free text fields using a military vocabulary.

These rules for interaction using a COTS speech recognition product such as IBM ViaVoice or Dragon NaturallySpeaking in a Log Entry form of the Lotus Notes CSS Log database would allow the user to say any of the following:

"location Brisbane"
"location"
"Brisbane"

Use of either the first or last utterances would result in Brisbane being selected in the location field. The second utterance would open a dialog to select a location. The user can then say *"Brisbane"* and the dialog will be closed. The above example is used throughout this report to describe the various techniques to integrate Dragon NaturallySpeaking and IBM ViaVoice with the Lotus Notes CSS Log database.

The above user interfaces have been developed with the traditional input modalities of keyboard and mouse together with speech. Multi-modal user interfaces such as these are called speech user interfaces. Winig and Proteau propose three different kinds of speech user interfaces; speech-enabled, speech-aware and speech-centric (1999). A speech-enabled interface is a user interface whose underlying software is unaware of the existence of speech input. A speech-aware interface is where speech is an additional interface. In this case extra software development is undertaken with the existing software of the user interface. A speech centric interface is a user interface where interaction is primarily via speech recognition and/or speech synthesis (Winig and Proteau, 1999). Speech user interfaces can also be divided into three categories; unconstrained natural language, command and control and dialog trees. The problem with natural language interfaces is the recognition performance is only improved by the development of a domain specific vocabulary. Rigid syntax constraining possible input is used by command and control interfaces to reduce complexity. Dialog trees use directed prompts and a restricted range of prompts to guide the user through the application (Rosenfeld et al, 2001). These have been used in speech-based telephony services. All of the speech user interfaces developed with the Lotus Notes CSS Log and the Java CSS Log have a mixture of unconstrained natural language and command and control. They can also be referred to as speech-aware interfaces.

2. Speech Integration with a Lotus Notes Command Support System Log Database

The Lotus Notes Command Support System Log database is used in the Joint Personnel Manpower and Logistics cell of the Deployable Joint Force Headquarters. The database has five forms; Log Entry, Action, Comment, Shift Handover and Situation Report. These forms are used to log events and action staff in the headquarters regarding any issues in the current operational climate. The Comment form is used to provide extra information or discussion relating to issues raised in Log Entry or Action forms. The Shift Handover and the Situation Report are used to summarise activities over the last 8 or 24 hours respectively. The Log Entry form is the most complex with 15 fixed fields and 3 free text fields. The Event, Details and Attachments fields accept free text and the others are fixed fields. An example of this form is shown below in Figure 1.

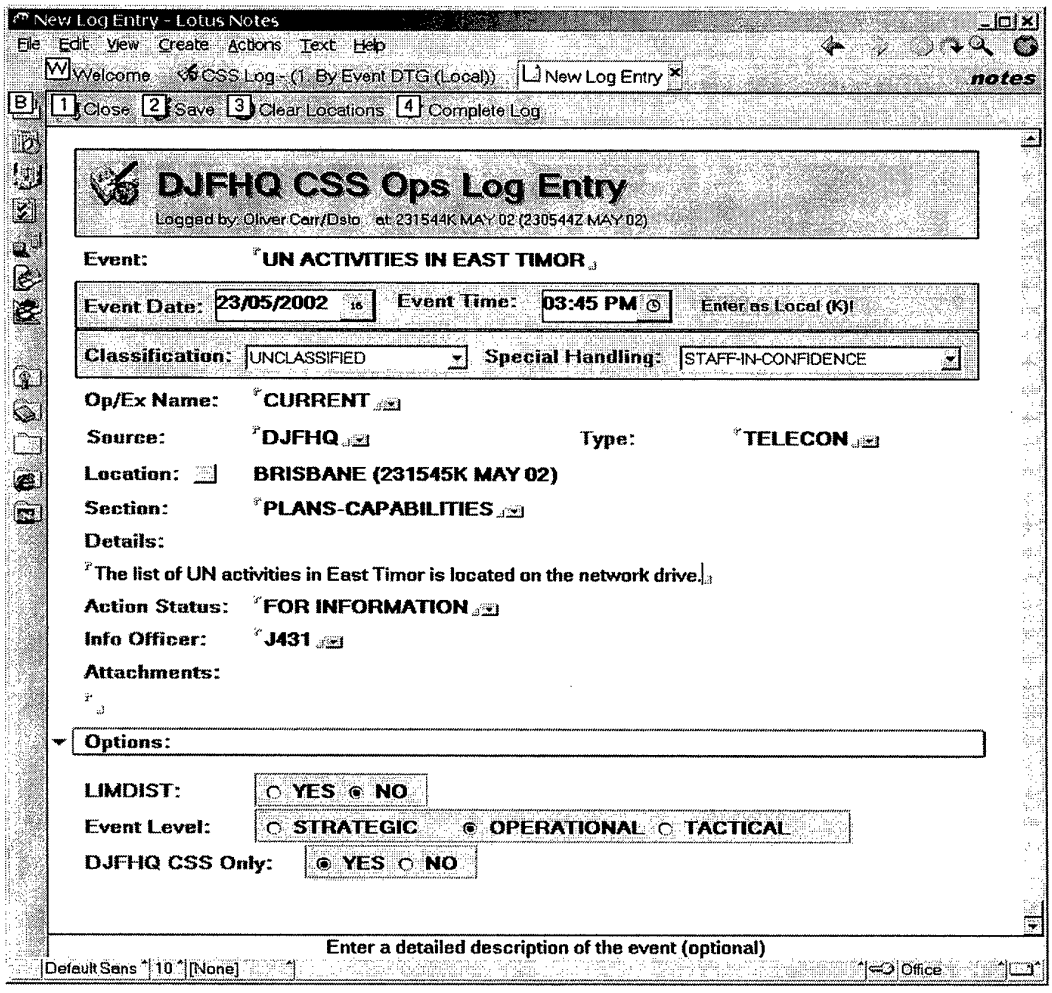


Figure 1 Lotus Notes Command Support System database Log Entry form

The Deployable Joint Force Headquarters has tasked us to investigate tools and techniques for incorporating speech recognition and synthesis with their Lotus Notes

CSS Log database. A speech-aware user interface to the Lotus Notes user interface was created using both Dragon NaturallySpeaking and IBM ViaVoice. Both speech recognisers have a method of integrating with an application via some sort of scripting language. These scripting languages are similar in function and syntax but differ in how they are programmed. Commands can be global or application specific. An application specific command is defined by the name of an application and the title of the current window. When an utterance is recognised, it is first tested against the current set of commands. Otherwise the utterance is regarded as dictation.

For the best possible integration of an automatic speech recogniser and a user interface, careful attention must be applied to the transaction design (Peckam, 1984). The transaction design for speech input into the Lotus Notes CSS Log allows for the user to fill out fixed fields in any order and dictate into free text fields. The only limitation is that the Lotus Notes CSS Log user interface only accepts a location in the Location field after the Event Date and Event Time fields have been entered. The transaction design of this speech user interface that mixes unconstrained natural language and command and control is shown below (Rosenfeld et al, 2001).

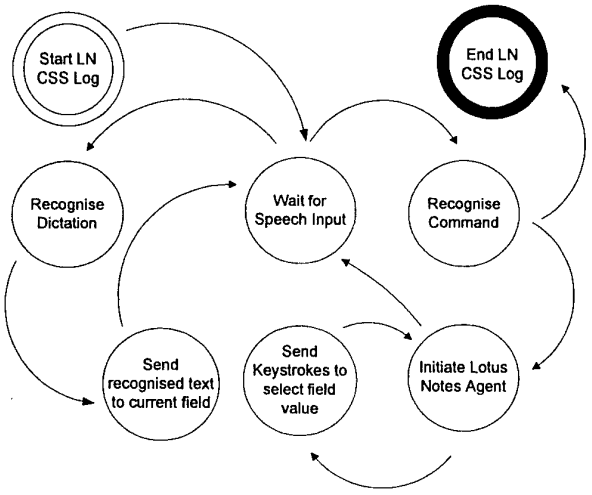


Figure 2 Lotus Notes CSS Log Speech User Interface Transaction Design

The Initiate Lotus Notes Agent state moves the cursor to the Details field, shown in Figure 1. For each command recognised, a series of *{Shift+Tab}* and *{Tab}* or other keyboard events could not be used to move to the appropriate field in the forms of the Lotus Notes CSS Log database. This was due to the mixture of fixed and free text fields in the forms such as the Log Entry as shown in Figure 1. A Lotus Notes Agent was added to the Lotus Notes CSS Log to create a common reference point for the commands written in the scripting languages. This Agent moves the cursor to the Details field and is called before and after every command. If a command is misrecognised, the recognised text is typed into the Details field. Otherwise, if a fixed field were in focus, the field's value would be changed. This extra Lotus Notes Agent added to the Lotus Notes CSS Log database is what makes the speech user interfaces presented in this section speech-aware and not speech-enabled.

2.1 Integrating Dragon NaturallySpeaking with Lotus Notes CSS Log Database User Interface Via Scripting Commands

The Dragon NaturallySpeaking Scripting Language is one method in which Dragon NaturallySpeaking can be used with a Windows application (Dragon Systems, 1998). The user can issue commands and dictate free text including punctuation. These commands can, for example, start Windows programs, operate menus and format text in Microsoft Word. All the scripting commands are defined for each window of an application in a text file. These commands can be created using an eight step Command Wizard or a text editor. Below is an example of a command to set the contents of the Location field in Figure 1.

```
MENU "NLNOTES" {
  STATE "New Log Entry - Lotus Notes" {
    COMMAND "location <location>" {
      SCRIPT {
        SendKeys "{Alt+a}+t"
        I = 2
        DO UNTIL I = 0
          SendKeys "{Shift+Tab}"
          I = I - 1
        LOOP
        SendKeys "{Space}"
        if _arg1 = "Brisbane" then SendKeys "b"
        if _arg1 = "Townsville" then SendKeys "tt"
        if _arg1 = "Darwin" then SendKeys "d"
        SendKeys "{Tab}{Tab}{Enter}"
      }
    }
    LIST "location" {
      "Brisbane"
      "Townsville"
      "Darwin"
    }
  }
}
```

Figure 3 Example of Dragon NaturallySpeaking Scripting Commands for the Location Rule

Speech synthesis can be incorporated in the commands to provide verbal instruction, feedback or assistance (Weinschenk and Barker, 2000). The gender, pitch, and speaking rate of the synthesised speech can be modified but are the same, for each user, for all commands and applications. The `Wait` function must be used after the `TTSPlayString` function, which performs the speech synthesis, to stop the flow of control for a predetermined number of milliseconds. This will prevent the microphone from accessing the sound card whilst the speech synthesis is being produced. The length of time in milliseconds passed as an argument to the `Wait` function is determined by trial and error for each `TTSPlayString` function call.

2.2 Integrating Dragon NaturallySpeaking with a Lotus Notes CSS Log Database via a Python-Based Macro System

In July 1999 Joel Gould, then Director of Emerging Technologies at Dragon Systems, released the first version of NatLink, a compatibility module for Dragon NaturallySpeaking (Gould, 1999a). This module enables users of NatLink to create

additional speech macros for Dragon NaturallySpeaking in a programming language called Python. This high-level object-oriented language is used as a scripting language and a rapid application development tool in many domains (Lutz, 2002). NatLink was developed to address the cumbersome macro development process for Dragon NaturallySpeaking Scripting Commands discussed above (Gould, 2001).

The Microsoft Speech API contains four APIs, Direct Speech Recognition API, Speech Services API, Voice Command API and Voice Dictation API (Microsoft Corporation, 2002). The Direct Speech Recognition API is a low level interface which controls the speech engine including loading speaker profiles, turning on and off the microphone, adding words to the vocabulary and creating grammars. Grammar objects are created to represent the allowable sequence of words to be recognised. When speech is recognised, result objects are created containing the words recognised and the corresponding grammars where they are defined. The grammars used in NatLink to define the macros are a small subset of those available in the Microsoft Speech API. Since NatLink is written in a combination of C++ and Python, it is able to reference directly the Microsoft SAPI (Gould, 2001).

For any Windows application, a Python source file can be written to define grammars and resultant actions. Figure 4 below is a simplified version of the source file written for the Lotus Notes CSS Log. For example, a grammar object is created for the Log Entry form, which contains three rules. Only one rule `<LogEntry>` is exported to the active vocabulary, which allows users to say, for example, *"location Brisbane"*. The square brackets denote optional words so the user could also say *"Brisbane"*.

```
import natlink
from natlinkutils import *

class ThisGrammar(GrammarBase):

  gramSpec = """
    <action> = ( 'action required' | 'for information');
    <Location> = ( Brisbane | Townsville | Rockhampton | Darwin );
    <LogEntry> exported = ( [location] <Location> |
                            [action status] <Action> );
      """

def initialize(self):
  self.load(self.gramSpec)

def gotBegin(self,moduleInfo):
  if matchWindow(moduleInfo,'nlnotes','New Log Entry - Lotus Notes'):
    self.deactivateAll()
    self.activate('LogEntry',noError=1)

def gotResults_Location(self,words,fullResults):
  natlink.playString('{Alt+a}t{Shift+Tab}{Shift+Tab}')
  if (words[0] == 'Brisbane'):
    natlink.playString('{Space}b{Tab}{Tab}{Enter}')
  elif (words[0] == 'Townsville'):
    natlink.playString('{Space}tt{Tab}{Tab}{Enter}')
  elif (words[0] == 'Darwin'):
    natlink.playString('{Space}d{Tab}{Tab}{Enter}')
```

Figure 4 Python Based Scripting Commands Location Rule Example

The `gotBegin` function is called when the application is in focus and the microphone is on. Depending on the particular window of the application, this function will activate the appropriate rules from the grammar object. For every rule in the grammar a function call is returned, `gotResults_Rule(self,words,fullResults)` where `Rule` is the name of the rule, `self` is a reference to the grammar class, `words` is the words recognised from the rule and `fullResults` contains the whole utterance (Gould, 2001). The `playstring` function mimics the `SendKeys` function used in the Dragon NaturallySpeaking Scripting Language. The code above for the `gotResults_Location` function is very similar to the Scripting Commands in Figure 3. However, the rules defined in NatLink grammars can be used within other rules and in different windows of an application.

2.3 Integrating IBM ViaVoice with a Lotus Notes CSS Log Database via Scripting Commands

There are two types of Scripting Commands available in IBM ViaVoice, Dictation Macros and Navigation Macros. As the name suggests Dictation Macros are scripting commands used when the user is dictating. For example if the user says *"exclamation mark"* during dictation the shriek character (!) will be substituted. The user can incorporate Dictation Commands at anytime during an utterance. Navigation Commands are isolated utterances that result in a combination of keystrokes being send to an application. These commands can be viewed via a tree structure corresponding to application, window and command in the Navigation Macro Creator which is shown in Figure 5. New commands can only be created in the Navigation Macro Wizard. The steps of the wizard define the name of the command, the application and the name of the window the command corresponds to, any lists to be used in the command, and the keystrokes to be sent. The wizard records the user typing the appropriate keystrokes into the application. A command can only be modified by deletion and re-creation in the wizard. All the commands or a subset of them can be imported and exported. However, the file format is binary and cannot be modified with a text editor.

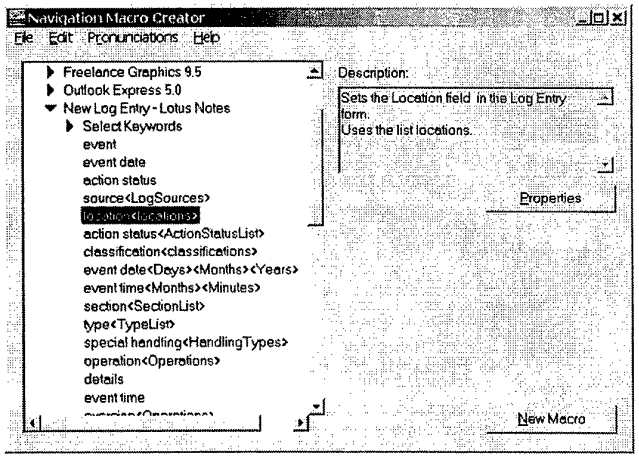


Figure 5 Example of IBM ViaVoice Navigation Macro Creator for Lotus Notes CSS Log

The `location<locations>` command, listed above in Figure 5 and shown in Figure 6, uses a custom list. This list, `<locations>` is defined as a list of words with a corresponding written text for each element. For example if the user says *"location Brisbane"* the letter *b* will replace the variable "locations" in Figure 6. However, it is not possible to create commands such as `<locations>` with no text before and no text after the list. The ViaVoice Marks Property allows speech synthesis or audio files to be played before and after a command is recognised.

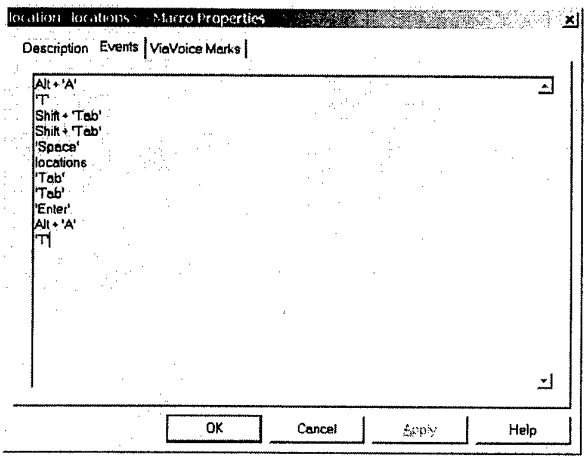


Figure 6 Example of a Scripting Command in IBM ViaVoice

2.4 Integrating IBM ViaVoice with a Lotus Notes CSS Log Database for Speech Synthesis

The Deployable Joint Force Headquarters have tasked us to investigate the use of speech synthesis for issuing reminders and informing users of new messages. IBM's Speech for Java implementation was used with IBM ViaVoice to provide the speech synthesis (IBM, 1998). Additional Lotus Notes Agents written in Java Script were developed and added to the Lotus Notes CSS Log database. This enables new database messages to be automatically read out to the user. The content of the synthesised speech was the title, classification and author of the new message. Normalisation was used to expand the short form of some acronyms to long form, to increase understanding of the spoken text (Weinschenk and Barker, 2000). For example `MAJ` would be expanded to `Major`. An experimental product called NotesBuddy was developed by IBM to provide the same features above and awareness of changes in Lotus' collaborative tool, Lotus Sametime (Lotus Software, 2001). NotesBuddy implements normalisation by providing users with a two column table to translate between written form and spoken form. A scheduled Lotus Notes Agent was incorporated into the Lotus Notes CSS Log database to allow text to be spoken to the user at a specified time or repeated at certain time intervals as a reminder. NotesBuddy does not have this feature (Lotus Software, 2001). In both products, the user was also provided with a properties dialog to change the gender, pitch and speaking rate of the synthesised speech.

3. Speech Integration with a Java Command Support System Log Database

A Java user interface called the Java CSS Log was developed to improve user interaction with the Lotus Notes CSS Log database. Java was chosen because Lotus Notes comes with Java classes for accessing and updating database information. Java is also an object oriented programming language, provides relatively easy to use GUI components and allows for flexible deployment options (Lewis and Loftus, 1998). The Java Swing package was used to provide the majority of GUI components used in the user interface. By developing the interface in Java, the Lotus Notes database information can be viewed easily inside a web browser.

The forms of the Lotus Notes CSS databases were redesigned incorporating improvements to the user interface. These improvements where provided by the operators of the Lotus Notes CSS Log database at the Deployable Joint Force Headquarters. The improvements are:

- Each field uses code to validate input. For example, the Event Time field will not accept alphabetic characters.
- Context sensitive help for the current field in focus can be obtained by pressing the F1 key or placing the mouse over the field.
- Greater visibility for all fields in the form.
- Common fields for all forms, Logged By, Logged At, Classification, Special Handling and Options, are placed in the same location.
- Selecting the same officer in the Shift Handover From and Shift Handover To fields of the Shift Handover form is not possible.

The Lotus Notes CSS Log database user interface requires the user to complete certain fields before others. For example, the Event Date and Time must be correctly filled in before Location information can be entered. When saving a Lotus Notes CSS Log database form, if more than one mandatory field is left blank, the user is prompted for each field's information, one at a time until all required elements are entered. The user is not alerted if there is one or more mandatory fields requiring information, and is only made aware of another error after hitting the save button again. When the save button is clicked in the Java CSS Log user interface, all mandatory fields requiring information are presented, thus saving the user multiple correction steps.

The transaction design for the speech recognition interface to the Java CSS Log incorporates two recognition modes, Command Mode and Dictate Mode. These modes were added so the user could not set the contents of a fixed field inadvertently whilst dictating. The Dictate Mode state applies to the free text fields, e.g. Event and Details in Figure 1. The user can say *"command mode"* to change the state from Dictate Mode to Command Mode. There are two types of commands for selecting field values and one type for navigating fields. Field values can be set by saying the field name and value or alternatively by saying just the value. For example, in Command Mode, the Classification field can be set to Unclassified by commands *"classification unclassified"* or *"unclassified"*. Similarly the command *"classification"* will move the cursor to the Classification field and present the user with a pull down list of classifications. The transaction design for speech recognition

interface to the Java CSS Log is shown below. Note the state transitions correspond to voice input, but keyboard and mouse input could result in different state transitions.

The speech user interface described by the transaction design below for the Java CSS Log is speech-aware. This interface has a greater level of "awareness" than the speech-aware user interfaces described in the previous section. By using Java to implement the GUI and the interface to the COTS speech recognition products a far greater level of control is available. For example, new words entered in dialogs of the Java GUI could also be automatically added to the vocabulary of the COTS speech recognition product.

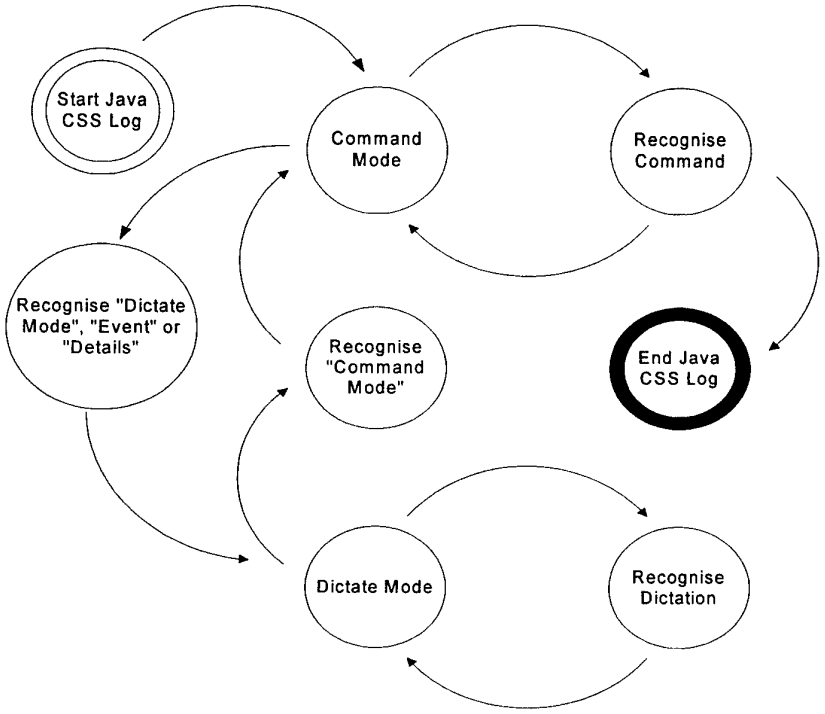


Figure 7 Transaction Design for Speech Input with Java CSS Log

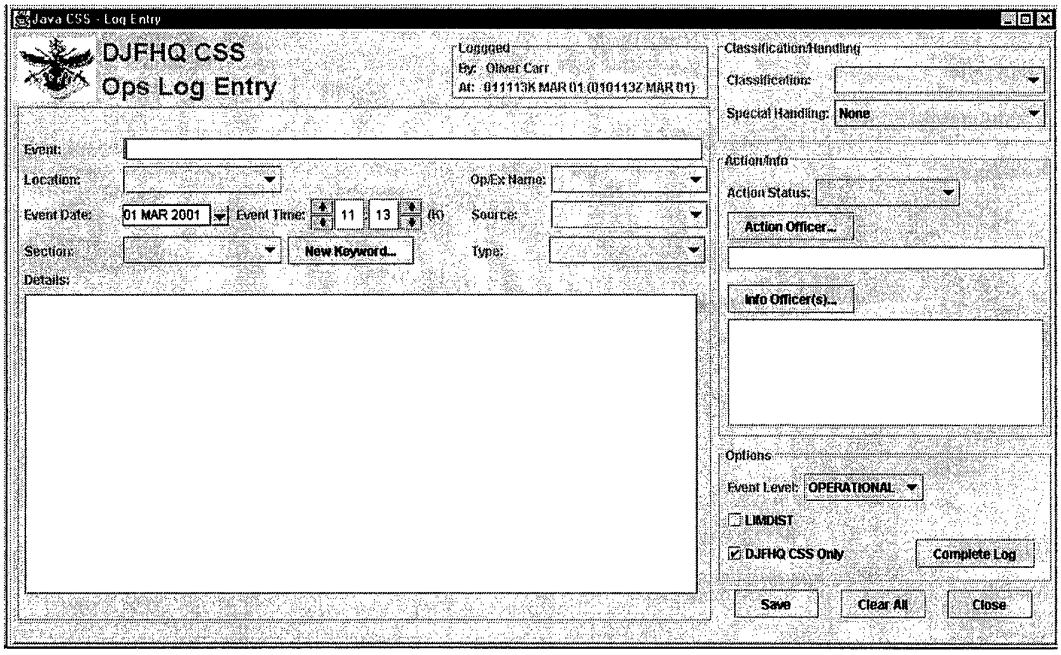


Figure 8 Java Command Support System Log Entry form

3.1 Dragon NaturallySpeaking Integrating with Java Command Support System Log Database

Dragon NaturallySpeaking and the Microsoft Speech API (SAPI) are both based on the Microsoft Common Object Model (COM). The Dragon NaturallySpeaking SDK is a set of ActiveX components, which use COM as the transport layer (Rogerson, 1997). The components cover the Engine Control, Dictation, Commands and Text-to-Speech aspects of Dragon NaturallySpeaking. The SAPI can be used by applications written in C++ to interface with Dragon NaturallySpeaking. However, there is no implementation of the Java Speech API or any other Java interface available for DNS (Sun Microsystems, 2002). J-Integra, Java-ActiveX bridge software, was used to generate Java wrapper classes for the Dragon NaturallySpeaking SDK COM components. The dictation-related DNS SDK ActiveX components could unfortunately not be used since they contained Microsoft Windows specific arguments, which are not supported by an operating system portable language such as Java. However, passing a specific constant to one of the other DNS SDK ActiveX components enabled dictation.

The voice command menus for the six forms and various dialogs of the Java CSS Log are set up at program initialisation. The `listSet` method is used to set up named lists (argument lists), each consisting of a finite set of strings. These lists are used in the rules to fill out the fixed fields in the forms of the Java CSS Log user interface. For example, the argument list `"<location_arg>"` consists of the strings `"Brisbane"`, `"Townsville"` and `"Darwin"`. These argument lists can be referenced within the definitions of any of the voice commands specified subsequently. For example, the `"Location"` command will be defined by the rule `"Location <location_arg>"`. The `menuCreate` function, shown in Figure 9, is used to create a voice menu. Only one menu is needed for an entire application. All the voice commands are added to a voice menu using the `add` function. The voice commands are differentiated in a voice

menu by a numeric identifier. For example to create the voice menu and add the "Location" rule the following function calls would be made:

```
menuCreate("Java CSS Log", "All Commands",
           DgnLanguageConstants.dgnlangUKEnglish,"",
           CREATEFLAGS.vcmdmc_CREATE_TEMP);
listSet(21,"location_arg","Brisbane / Townsville / Darwin");
add(21,"location <location_arg>","Location Command",
    "Java CSS Log", null, null);
```

Figure 9 Java Wrapper Function Calls for Location Rule

The voice command menu (or active vocabulary) could have been changed dynamically between the various Java CSS Log forms. However, it proved programmatically easier to keep the voice command menu static and monitor the current application state. For each voice command recognised, the appropriate field selection is done depending on the current Java CSS Log form. Two extra steps were added to the transaction design for the Java CSS Log to toggle between Command Mode and Dictate Mode. The user can switch between these two modes by invoking the commands, *"command mode"* or *"dictate mode"* respectively. Commands of the same name were added to the voice command menu. One limitation of a static vocabulary occurs when fields or buttons with the same name appear on different forms. For example, every form contained a Close button with a corresponding Close command. These voice commands have the same spoken text but different command identifiers. It was found empirically that the DNS Engine Control ActiveX object reports the numerically smallest voice command identifier when a particular multiply allocated voice command is recognised. The solution to this problem is to test which dialog is visible to the user and initiate the appropriate Close button click event programmatically.

3.2 IBM ViaVoice Integration with Java Command Support System Log User Interface

Speech recognisers use grammars to improve recognition accuracy and response time. The Java Speech Grammar Format (JSGF) defines the grammars used in the Java Speech API (Sun Microsystems, 1998a). There are different grammar types for command and control, and dictation. The JSGF describes a command and control grammar as a rule grammar. A rule is a combination of spoken text and references to other rules. A grammar is a set of these rules that define what can be spoken. These grammars are defined in a text file and parsed by the JSAPI function `loadJSGF`. The Speech for Java implementation of the Java Speech API was used to provide a software interface between Java and IBM ViaVoice (IBM, 1998). An example of a grammar containing the rule grammar for the location field of the Java CSS Log Entry form is shown in Figure 10.

Two rule names are used to define the two possibilities of selecting a location by voice. The rule `alone_location_arg` allows the user to simply say one of the locations defined in the list `location_arg`. Alternatively, the rule `location` allows the user to say for example *"location Brisbane"* or *"location"* which brings up the Location dialog. Rule grammars can also be defined within Java by using Java Speech API methods. The words in the braces, for example `{Darwin}`, are called labels and are returned as a result object upon successful recognition. An example to set the Location rule

grammars is shown below in Figure 11. The Boolean parameter to the `setRule` function defines whether the rule is to be used in another rule or exported to the active vocabulary. The `location_arg` rule is used within the `location` rule, which allows the user to say, for example, *"location Darwin"*.

```
grammar JavaCSSLog;

<location_arg> = Brisbane {Brisbane} | Townsville {Townsville}
                 | Darwin {Darwin};

public <alone_location_arg> = <location_arg> {Location};

public <location> = (Location <location_arg> | Location {Location})
                    {Location};
```

Figure 10 Example of Java Speech Grammar Format definition of Location rule grammar

```
import javax.speech.*;
import javax.speech.recognition.*;

Rule LocationArg, AloneLocation, Location;

LocationArg = ruleGrammar.ruleForJSGF("Brisbane {Brisbane} | Townsville
                                       {Townsville} | Darwin {Darwin}");
ruleGrammar.setRule("location_arg", LocationArg, false);

AloneLocationRule = ruleGrammar.ruleForJSGF("<location_arg> {Location}");

ruleGrammar.setRule("alone_location_arg", AloneLocationRule, true);

LocationRule = ruleGrammar.ruleForJSGF("(Location <location_arg> |
                                        Location {Location}) {Location}");
ruleGrammar.setRule("location", LocationRule, true);
```

Figure 11 Examples of Java Speech API Function Calls for Location Rule Grammars

The user can say the commands *"Event"* or *"Details"* to move focus to those free text fields. When this occurs the rule grammar is disabled and a dictation grammar enabled. The dictation grammar defines rules for formatting text, such as *"Select All"*, and handles presentation hints for spoken text. For example if the user said *"3 point 1 4"* the presentation hints would change the utterance to *"3.14"*. The dictation grammar for the Java CSS Log also defines the command *"command mode"* to re-enable the rule grammar.

3.3 Java Speech API Implementation for Dragon NaturallySpeaking

There is currently no implementation of the Java Speech API for Dragon NaturallySpeaking (Sun Microsystems, 2002). An effective implementation will allow developers and researchers to undertake comparative studies of Dragon NaturallySpeaking and IBM ViaVoice with a Java application via the Java Speech API. Java data structures were implemented to translate the Java Speech API methods to the Java methods that interface to the Dragon NaturallySpeaking SDK. As shown in Figure 12 below, J-Integra was used to provide a software interface between Java and the ActiveX components supplied with the Dragon NaturallySpeaking SDK. The Java CSS Log user interface has been successfully integrated with Dragon NaturallySpeaking using this method. Future work will

combine the two implementations of the Java Speech API, the IBM Speech for Java implementation and our in-house developed implementation using J-Integra. This will enable the application to automatically detect and work with whichever speech recogniser is installed and being used by the user.

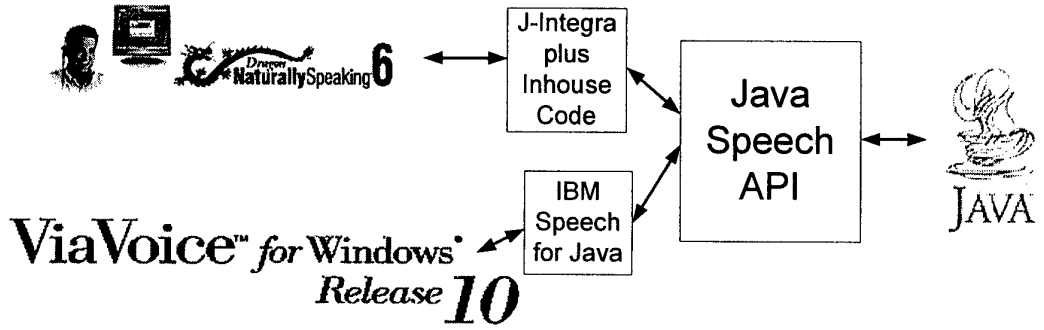


Figure 12 Component Level Diagram of Interfacing Java Software with Dragon NaturallySpeaking and IBM ViaVoice Using Java Speech API Implementations

4. Developing a Military Vocabulary for Dragon NaturallySpeaking and IBM ViaVoice

In every domain acronyms and abbreviations are used in both written and spoken communication (Larkey et al, 2000). The Australian Defence Force uses a document called ADFP 123 Acronyms and Abbreviations to define over 4,000 acronyms and abbreviations. Dragon NaturallySpeaking and IBM ViaVoice contain vocabularies of 250,000 and 160,000 words respectively (ScanSoft, 2002; IBM, 2002). In DNS the user can modify the original vocabulary with Vocabulary Editor and Vocabulary Builder tools provided. The Vocabulary Editor, shown in Figure 13, lists the words in the vocabulary by their written and spoken forms. The spoken form can be omitted if it is the same as the written form. Similar functionality can be achieved with IBM ViaVoice by using the Dictation Macro Editor shown in Figure 14. The Vocabulary Builder allows the user to add a list of words as phrases, by automatically generating pronunciations, and also scans Word documents for new words. IBM ViaVoice has a similar tool called Analyse My Documents. These tools do not allow the user to add large amounts of words with written and spoken forms easily.

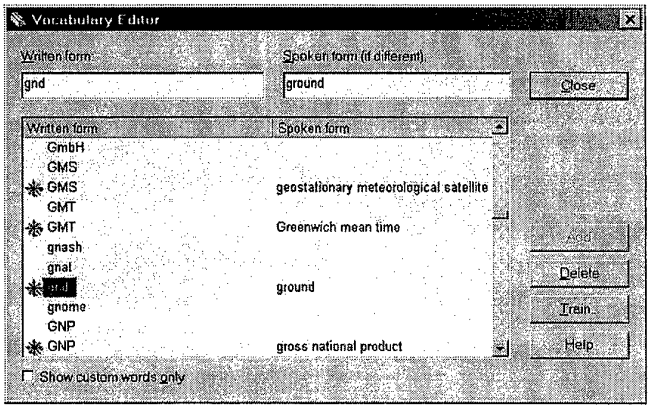


Figure 13 Dragon NaturallySpeaking Vocabulary Editor

The Unofficial Dragon NaturallySpeaking web site provides a tool called PutWords, which allows words to be added to a user's vocabulary from a text file (Gould, 1999b). The format of the text file is `<written form>\<spoken form>` per line. For example the acronym ADF would be added as follows: `ADF\Australian Defence Force`. Another pronunciation such as *"a d f"* could also be added, i.e. `ADF\a d f`. The 4000 acronyms mentioned above can be added to a users vocabulary using PutWords. The generation of this text file by hand is very tedious.

To address this a prototype Acronym Manager was developed to allow the user to add to their Dragon NaturallySpeaking vocabulary a selection of over 40,000 acronyms, which have been provided by the Defence Materiel Organisation. The aim of the Acronym Manager was to:

- Load text files set out in the `<written form>\<spoken form>` format.
- Automatically create different pronunciations for acronyms.
- Find different expansions of acronyms from the Internet.
- Create groups of acronyms for other users.

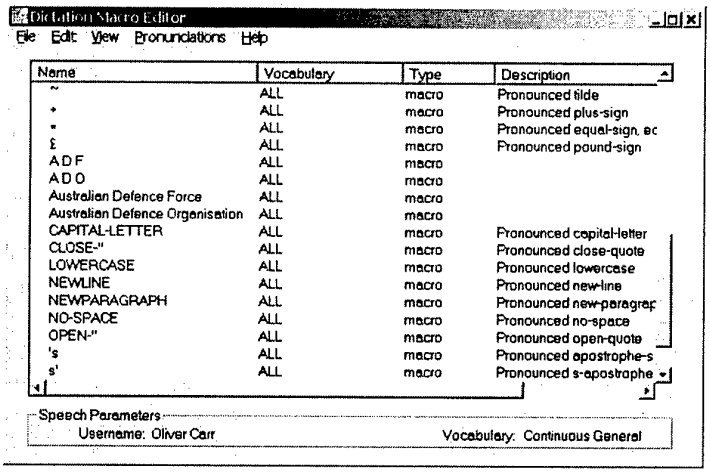


Figure 14 IBM ViaVoice Dictation Macro Editor

Currently the acronyms and abbreviations are separated into text files for each letter of the alphabet. For example, all the acronyms and abbreviations beginning with the letter 'a' or 'A' are in one text file. There are many different types of acronyms that can be pronounced in various ways. For example ANZUS stands for Australia New Zealand United States, which could be pronounced, *"a n z u s"*, *"ann zuss"* or *"Australia New Zealand United States"*. Some acronyms also have another acronym incorporated in them. For example, ARF stands for ASEAN Regional Forum where ASEAN stands for Association of South East Asian nations. The user could pronounce this acronym *"Association of South East Asian nations regional forum"*, *"a z ian regional forum"* or *"a r f"*. After the user has selected the acronyms to add to their vocabulary, a dialog shown below in Figure 15, allows the user to select various types of pronunciations. Many acronyms have the same short form and so for acronyms like ARF above, sometimes it is not clear what the expanded form should be. The Acronym Manager addressed this by providing the user with a dialog to select the appropriate long form, which is shown below in Figure 16. The user is also able to use one of many Internet directories such as Acronym Finder to find long forms of acronyms (Acronym Finder, 2002).

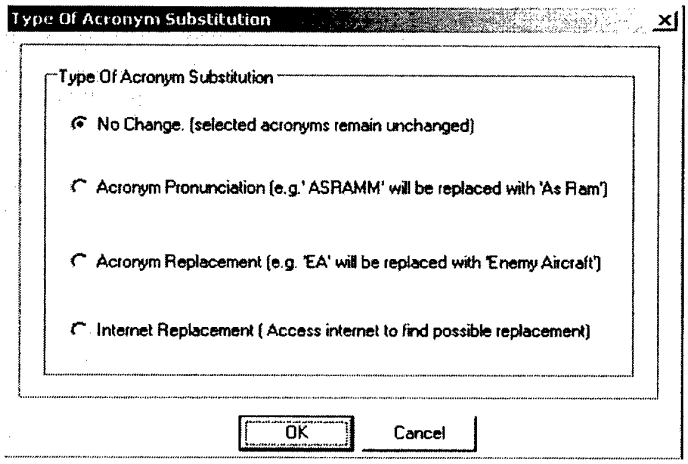


Figure 15 Type of Acronym Substitution Dialog of the Acronym Manager

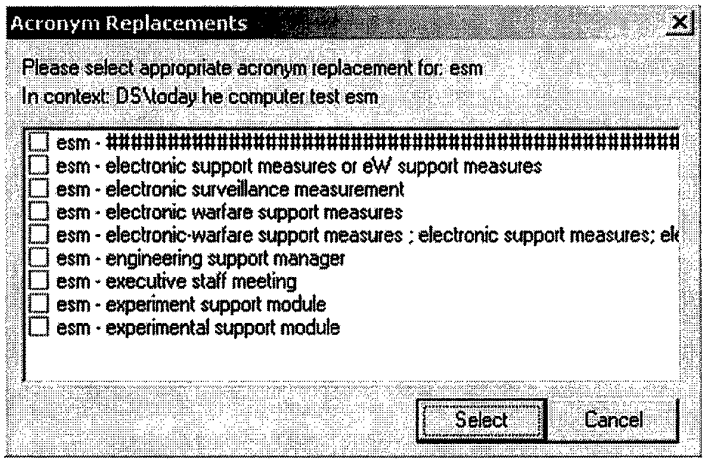


Figure 16 Acronym Replacements Dialog of the Acronym Manager

IBM ViaVoice provides users with tools to find new words in documents and improve the language model by analysing previously written documents. The user can add new words to the vocabulary by creating Dictation Macros. These macros are a simple version of the Navigation Macros and provide a correspondence between spoken and written text. The user can create Dictation Macros using a wizard and import or export existing macros in a proprietary file format. The list of words added to the vocabulary can be viewed and deleted in the Vocabulary Manager shown in Figure 17. Note that only the spoken forms are shown and not the written form, unlike with the Dragon NaturallySpeaking Vocabulary Editor in Figure 13. The list of 4,000 Australian Defence Force acronyms were added to the IBM ViaVoice vocabulary by using NatLink and Python to read in the text file and initiate the appropriate keystrokes. The vocabulary can now been exported using the Dictation Macro Editor and imported by other users and installations.

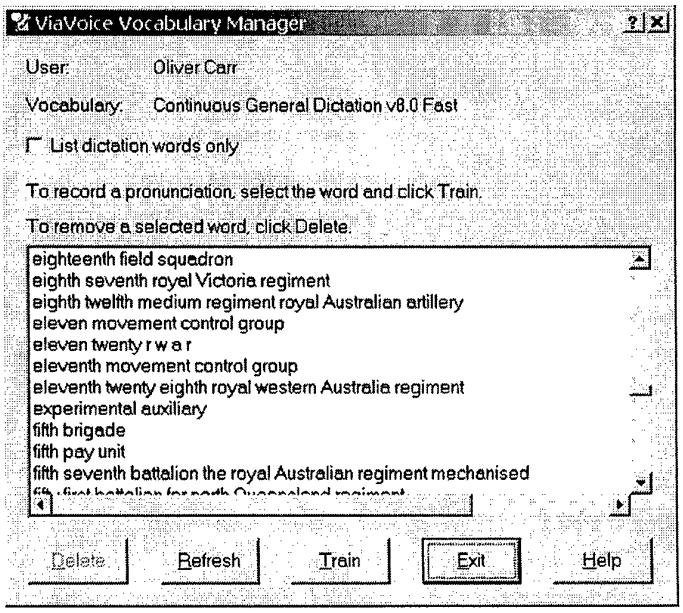


Figure 17 IBM ViaVoice Vocabulary Manager

5. Evaluation of Dragon NaturallySpeaking and IBM ViaVoice for Command and Control

Assessment methods developed recently for speech recognition involve a combination of system-based approaches and performance-based techniques (Pallett and Fourcin, 1995). Before the Australian Defence Force adopts a COTS speech recognition product it is important to evaluate its suitability to the eventual military system. Barbarello and Kasian from the Research, Development and Engineering Centre of the US Army Communications-Electronics Command from their experience with COTS materiel solutions advocate the above approach (2000). It has been suggested that user interface acceptance and performance is improved by adopting a multimodal approach (Grasso et al, 1998). Many evaluations of automatic speech recognisers in both the commercial and research domain, concentrate on performance in terms of recognition accuracy.

An evaluation of the speed of the interaction techniques in using Dragon NaturallySpeaking and IBM ViaVoice for command and control of an application was undertaken. The application chosen was the Lotus Notes Command Support System database and in particular, the Log Entry form. A basic comparison of both products is shown below in Table 1. Both products support Lotus Notes "out of the box" but only provide the user with field navigation and basic Window management commands. The Andrea microphone supplied with IBM ViaVoice is a superior microphone than the Telex microphone supplied with Dragon NaturallySpeaking. Both speech recognisers improve in terms of accuracy with additional training (McEvoy, 2000a; 2000b). The cost of the two products can also vary depending on volume license agreements and the product family chosen from. For example, Dragon NaturallySpeaking Preferred is around half the price of Dragon NaturallySpeaking Professional. Dragon NaturallySpeaking Preferred has fewer features than Professional but the same internal speech recognition engine. As reported by Gould and verified in our laboratory, the NatLink compatibility module for Python and the PutWords programs both work with Dragon NaturallySpeaking Preferred (1999a; 2001).

Table 1 Basic Comparison of Dragon NaturallySpeaking Pro 6 versus IBM ViaVoice Pro USB 9

	Dragon NaturallySpeaking Pro 6	IBM ViaVoice Pro USB 9
Min. PC Requirements	500 Mhz CPU with 256 MB RAM	300Mhz CPU with 96 MB RAM
Required HD Space	300 Mb plus 60 MB/User	510 Mb plus 2 MB/User
USB Input and Adapter	Yes	Yes
Microphone Supplied	Telex H-841	Andrea NC-61
Unit Cost	A$1600	A$300
User Training Time (min)	20 (inc. installation) [2]	30 (inc. installation) [2]
Accuracy	95%[2]	92%[3]

The execution time of a Scripting Command in the Log Entry form was measured after an utterance is recognised. Lotus Script code was written in Lotus Domino Designer to time stamp the initiation of the Lotus Notes Agent mentioned in the

[2] Keizer, 2002
[3] Keizer, 2001

transaction design for Scripting Commands in Figure 2. Therefore we measured the difference in time from the first Scripting Command to the last command in the script. The completion of a field entry was also time stamped by writing Lotus Script code in the field's Exiting method. In this new Evaluation Log Entry form, each time stamp was added to a free text field where each line consisted of the field name, field value and the number of seconds since midnight. Figure 18 shows the resultant time stamps of a user saying *"location Brisbane"* using IBM ViaVoice in an Evaluation Log Entry form of the Lotus Notes CSS Log database.

Each speech integration technique using Scripting Commands for both IBM ViaVoice and Dragon NaturallySpeaking was used to fill out the Evaluation Log Entry form. Both products were used with the same 4,000 word military vocabulary. The Scripting Command techniques considered were the Dragon NaturallySpeaking Scripting Language, NatLink and IBM ViaVoice Navigation Macros. For each technique, the Evaluation Log Entry form was filled out 20 times with slightly different data for all the fields each time. One user conducted the experiment, as it was assumed the timing technique was user independent since recognition speed was not measured. The PC system used had the following configuration; AMD 1 GHz CPU, 512 MB of RAM, Creative Labs PCI 128 Sound Card, Windows 2000 Professional operating system and an Andrea NC-61 microphone used with both Dragon NaturallySpeaking and IBM ViaVoice.

```
Event Entering,  ,60158.49
Event Exiting,  ,60163.98
Alt a t pressed: , 60170.99
Details Exiting,  ,60172
Section Exiting,  ,60172.17
Location Exiting, BRISBANE, 60174.07
Alt a t pressed: , 60174.91
```

Figure 18 Time Stamps for Location Scripting Command from Lotus Notes CSS Log

The three integration techniques tested were the IBM Navigation Macros, the Dragon NaturallySpeaking Python-based Macro System and the Dragon NaturallySpeaking Macro Language. These are graphed in Figure 19 below as IBM, DNS Python and DNS Macro, respectively. The results show the IBM Navigation Macro method had the maximum average field completion time of 6.65 seconds, a minimum of 3.9 seconds and a standard deviation of 0.74 seconds. The DNS Python and DNS Macro techniques produced very similar results to each other. The difference in average field completion time for these two methods is shown below in Figure 20. The maximum average field completion time for the DNS Python technique was only 0.25 seconds, a minimum of 0.20 seconds and a standard deviation of 0.01 seconds. The DNS Macro technique had a maximum of 0.28 seconds, a minimum of 0.22 seconds and a standard deviation of 0.02 seconds. Figure 20 also shows that in two out of the 20 trials, the DNS Macro technique was quicker than the DNS Python technique. The modulus of the difference of the average field completion times of the DNS Macro and the DNS Python techniques was 0.03 seconds.

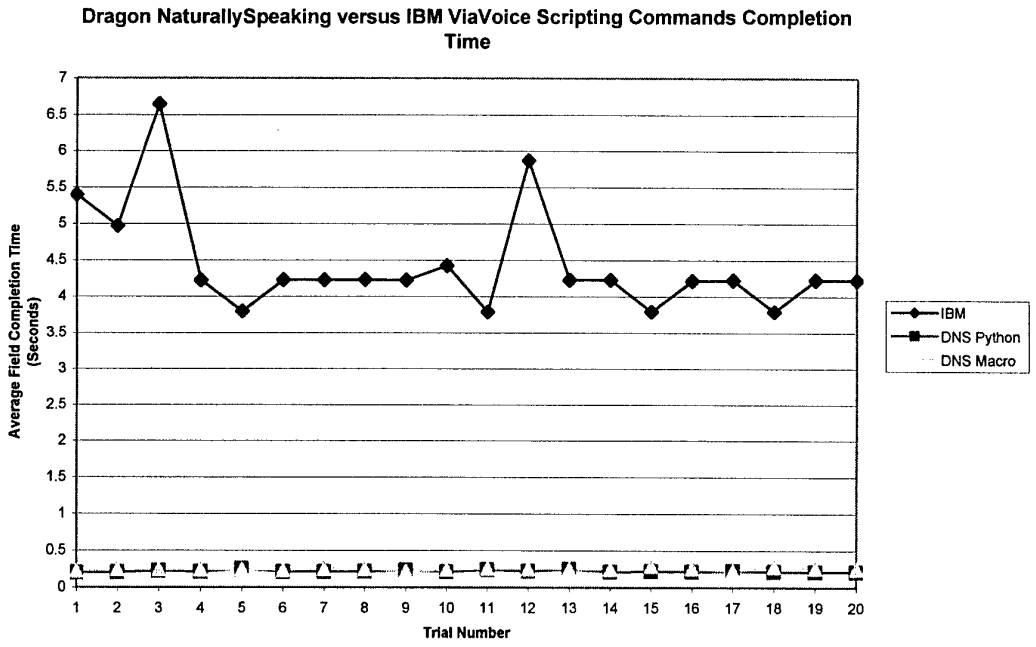


Figure 19 Average Field Completion Time for Dragon NaturallySpeaking versus IBM ViaVoice Scripting Commands Techniques.

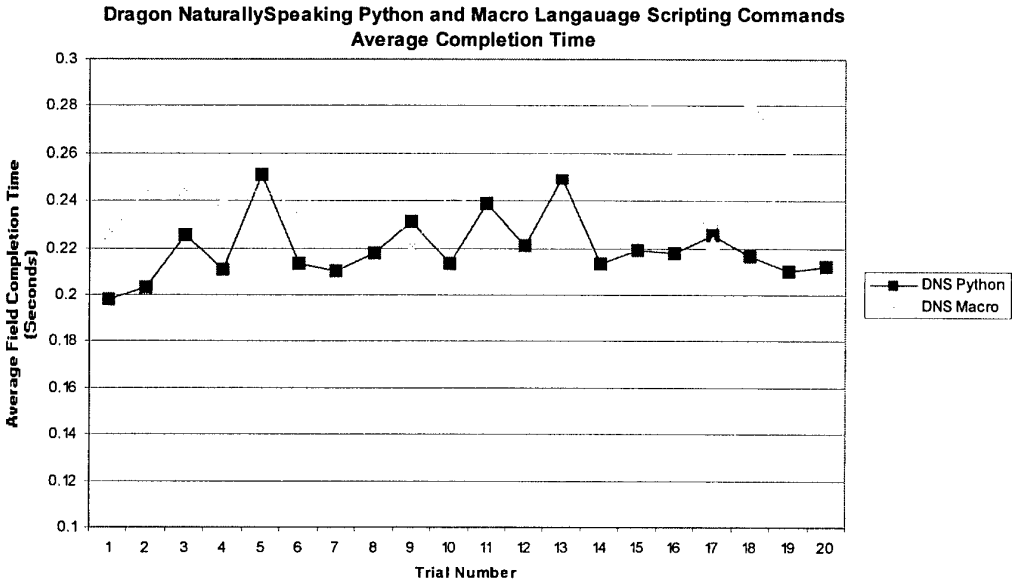


Figure 20 Average Field Completion Time for two Dragon NaturallySpeaking Scripting Command Techniques, Python-based system and Macro Language.

The Deployable Joint Force Headquarters has also provided four other Lotus Notes databases. These databases have an almost identical structure but contain different field values pertaining to a corresponding branch of the headquarters. The Dragon NaturallySpeaking Macro Language is a very simplified version of the Visual Basic programming language. The macros are defined in a hierarchy with Application, Window and Command levels. The Window level refers to the name of a window of

an application. For example in Word a new document has the title of "Document1 - Microsoft Word". The scope of declared variables is only at the Command level, i.e. there are no global variables. Therefore no code reuse is possible for the common fields contained in the four Log Entry forms of the four Lotus Notes databases. The NatLink method can be easily modified for other forms and databases. This was achieved by using global variables to indicate which form and database is in focus. Extra conditional statements were written in the functions such as `gotResults_Location`. By contrast, the Dragon NaturallySpeaking Macro Language required an extra function to recognise locations for each Log Entry form in the four databases. The number of lines required to provide the same speech input capability for all forms in the four Lotus Notes databases was 10,111 for the Dragon NaturallySpeaking Macro Language technique and 818 for the NatLink Python based technique. Therefore a Python-based Scripting Commands technique using NatLink would be far easier to maintain for support staff.

6. Discussion

Event languages, such as the various Scripting Command languages discussed above for Dragon NaturallySpeaking and IBM ViaVoice, may not map as well for speech user interfaces as Visual Basic did for Windows user interfaces (Myers et al, 2000). The Scripting Command methods described define a set of voice commands for a particular window of an application. The resultant actions of these commands mimic keystrokes that are sent to the application. There is a difference between these methods in how these voice commands are developed and modified. There is also a difference in the speed with which these methods respond once the command is recognised. IBM ViaVoice was found to be significantly slower in responding to commands than Dragon NaturallySpeaking. The Python-based technique was marginally faster than the Macro Language technique for Dragon NaturallySpeaking. Dragon NaturallySpeaking Professional version 6 allows for Visual Basic scripts to be used for Scripting Commands but we assume these would be as fast as those developed using the Dragon NaturallySpeaking Macro Language. The speed of the two Dragon NaturallySpeaking Scripting Command techniques is so fast that for the longer commands of more than four words, the Speed vs. Accuracy setting cannot be set to Fastest. Otherwise the natural pauses in spoken utterances can cause misrecognition errors. IBM ViaVoice is several percent more accurate for dictation than Dragon NaturallySpeaking (Keizer, 2001; Keizer, 2002).

The Python-based Macro System using NatLink is a more efficient method of developing Scripting Commands for an application. However, NatLink requires a good understanding of the Python programming language, which may be too costly for support staff to be trained in. Scripting languages such as Python are at first designed for non-professional programmers, but inevitably the full range of syntactic control structures are developed (Myers et al, 2000). The Vocola voice command language has been developed using NatLink to provide a more concise and easy to use language (Mohr, 2002). The Scripting Commands developed using NatLink could in the future be translated to Vocola. This would allow support staff to maintain the Scripting Commands without having to learn a complex programming language.

Many Windows-based applications share the same dialog interaction techniques. Automatic techniques for developing user interfaces at a high-level with details of implementation provided by the system have shown significant promise but have not found wide acceptance (Myers et al, 2000). Speech user interfaces developed for many different applications with similar transaction design principles are called Universal Speech Interfaces (Rosenfeld et al, 2001). High-level languages such as Elephant 2000 could be used to implement the transaction design described above for the Lotus Notes and Java versions of the CSS Log database (McCarthy, 1998). This approach would automatically generate the software to provide the interface between the chosen application and speech recogniser. The language could be used to automatically develop speech user interfaces for any Windows-based applications using Dragon NaturallySpeaking or IBM ViaVoice by automatically enabling command and control or unconstrained natural language speech input. Languages such as Elephant 2000 could also be used to manage the military vocabulary needed for different applications and users. Future work will categorise the 40,000 acronyms and abbreviations provided by the Defence Materiel Organisation into a Lotus Notes

database. The speech user interface for an application could use Dragon NaturallySpeaking or IBM ViaVoice depending on the mixture of command and control and unconstrained natural language speech input required. For example, a legal unit would use a COTS speech recognition product with a legal vocabulary for dictation. Alternatively, an operator in a Logistics, Personnel and Manpower unit would require command and control for filling out fixed fields of a Lotus Notes database and dictation for entering free text. In these two respective cases IBM ViaVoice and Dragon NaturallySpeaking would be recommended.

Most of the current user interfaces assume that they have the users full attention (Myers et al, 2000). During high operational tempo a military headquarters can be swamped with information, requests and administrative tasks. The speech synthesis we developed with IBM ViaVoice and incorporated into the Lotus Notes CSS log database was designed to notify users of certain messages arriving. This auditory notification was designed for users who could be busy with other tasks such as collaborative planning, talking on the telephone or to a colleague. However, speech output is sometimes difficult to understand (Yankelovich, 1998). A context aware normalisation of acronyms, from short form to long form, contained in the spoken messages, will aid better understanding. The speech user interface incorporating speech synthesis allowed the user to change the gender, pitch and speaking rate of the auditory output. There is evidence from experiments undertaken by Nass and Gong that the gender of synthesised speech can be changed depending on the gender of the listener, to make the output subjectively more convincing (2000). However, it is still an open question whether or not the user should be able to change the properties of the synthesised voice (Nass and Gong, 2000).

7. Conclusion

This report described two taxonomies of speech user interfaces for applications. The development of speech user interfaces for a Lotus Notes Command Support System Log database and a Java Command Support System Log database was shown together with a transaction design for both applications. Several techniques to develop speech user interfaces for Windows-based applications are described. The development of a military vocabulary was described for both Dragon NaturallySpeaking and IBM ViaVoice. We evaluated the speed of the speech-aware user interfaces developed for Lotus Notes using both Dragon NaturallySpeaking and IBM ViaVoice. Dragon NaturallySpeaking was found to respond to commands faster than IBM ViaVoice. However, IBM ViaVoice has a slightly better recognition accuracy than Dragon NaturallySpeaking. The design of a speech user interface for an application could use either Dragon NaturallySpeaking or IBM ViaVoice depending on the capabilities of the two products, cost of licences, development time and maintenance requirements. The use of high-level languages describing the transaction design could in the future automatically integrate an application with any given COTS speech recognition and synthesis product. The integration of speech recognition, speech synthesis and artificial intelligence technology could in the future result in a computer that understands speech and communicates as humans do (Lai, 2001).

8. Acknowledgements

The work documented in this report is funded through a DSTO task JNT 01/092 entitled "Human Computer Interaction Research, including Speech and Natural Language Approaches, for Command Support" managed by Dr Ahmad Hashemi-Sakhtsari and sponsored by Commander Deployable Joint Force Headquarters, MAJGEN Mark Evans and Head Knowledge Systems, RADM Peter Clarke. Without the fantastic support and interaction with Dr Ken Skinner and the staff at the Deployable Joint Force Headquarters, this work would not have been possible. Sven Wiener from Scandia Technologies provided most of the Java programming on contract. Anslem Teh from Flinders University completed the Java aspects whilst on work experience at Human Systems Integration Group, Command and Control Division, DSTO. Robert Dimasi from Flinders University started the task of managing acronyms and vocabularies for the COTS speech recognition software as a work experience project. Without the guidance from Ahmad Hashemi-Sakhtsariand his comments on draft reports and the enthusiasm of Anselm The, Robert Dimasi and the staff at Deployable Joint Force Headquarters this work would not have developed to its current level.

9. References

Acronym Finder (2002), Mountain Data Systems, http://www.acronymfinder.com

Barbarello, J.J., Kasian, W. (2000), "United States Army Commercial Off-The-Shelf (COTS) Experience - The Promises and Realities", Proc. Commercial Off-The-Shelf Products in Defence Applications - The Ruthless Pursuit of COTS, Information Systems Technology Panel Symposium, Research and Technology Organisation, NATO, December.

Behr, M. (2002), "Your Technology Future, According to Bill", PC Magazine, Zdiff Media Inc., June, http://www.pcmag.com/print_article/0,3048,a=27990,00.asp.

Dragon Systems (1998), "Dragon NaturallySpeaking Creating Voice Commands", August.

Fournery, P., Sorensen, U. (2000), "Lotus White Paper on COTS for Military Crisis Applications", Information Systems Technology Symposium on Commercial Off-the-Shelf Products in Defence Applications - The Ruthless Pursuit of COTS, RTO MP-48, Research and Technology Organisation, NATO, Belgium, April.
http://www.rta.nato.int/Activities.asp?Panel=IST&pg=2.

Gould, J. (1999a), "Python Macro System",
http://www.synapseadaptive.com/joel/PythonMacroSystem.html.

Gould, J. (1999b), "NaturallySpeaking Unofficial Information Pages",
http://www.synapseadaptive.com/joel/.

Gould, J. (2001), "Implementation and Acceptance of NatLink, a Python-Based Macro System for Dragon NaturallySpeaking", The Ninth International Python Conference, March 5th-8th, California, http://www.python9.org/p9-cdrom/15/index.htm.

Grasso, M., Ebert, D.S., Finn, T.W. (1998), "The Integrality of Speech in Multimodal Interfaces", ACM Transactions on Computer-Human Interaction, pp. 303-325, Vol. 5, No. 4, December.

IBM (1998), "Speech for Java", March,
http://www.alphaworks.ibm.com/tech/speech.

IBM (2002), "IBM ViaVoice for Windows Pro USB Edition Release 9",
http://www-3.ibm.com/software/speech/desktop/w9-pro.html.

Intrinsyc Software (2001), "J-Integra", Vancouver, British Columbia, Canada,
http://www.intrinsyc.com/products/j-integra.

Keizer, G. (2002), "Dragon NaturallySpeaking 6.0", CNET Networks Inc., April,
http://www.zdnet.com/supercenter/stories/review/0,12070,560820,00.html.

Keizer, G. (2001), "IBM ViaVoice Pro USB Edition 9.0 Review", CNET Networks Inc., October,
http://www.zdnet.com/supercenter/stories/review/0,12070,538007,00.html.

Kotmel, A. (2000), "Microsoft SAPI Audio Objects and the Dragon NaturallySpeaking SDK", Dragon Systems.

Lai, J. (2001), "When Computers Speak, Hear, and Understand", Communications of the ACM, pp. 66-67, Vol. 44, No. 3, March.

Larkey, L. S., Ogilvie, P., Price, M. A., Tamilio, B. (2000), "Acrophile: An Automated Acronym Extractor and Server", Proceedings of the 5th ACM Conference on Digital Libraries, pp. 205-214, ACM Press, San Antonio, Texas, U.S.A., June.

Lewis, J. and Loftus, W. (1998), "Java Software Solutions: Foundations of Program Design", Addison-Wesley Longman, Inc.

Lotus Software (2001), "NotesBuddy", alphaWorks, IBM Corporation, January,
http://www.alphaworks.ibm.com/tech/notesbuddy.

Lutz, M. (1996), "Programming Python", O'Reilly & Associates, Inc.

Maple, T. (2001), "Lotus & Defence - Lotus Software in the Australian Defence Organisation", Lotus Products in Defence, IBM, August.

McCarthy, J. (1998), "Elephant 2000: A Programming Language Based on Speech Acts", Stanford University,
http://www-formal.stanford.edu/jmc/elephant/elephant.html.

McEvoy, A. (2000a), "Battle of the Brands - Dragon NaturallySpeaking Preferred 5, ViaVoice for Windows Pro Edition compared", PC World Communication Inc., San Francisco, California, U.S.A., November,
http://www.pcworld.com/reviews/article.asp?aid=18500.

McEvoy, A. (2000b), "Rev your Speech Engines with IBM and L&H - New ViaVoice and NaturallySpeaking packages got us talking", PC World Communication Inc., San Francisco, California, U.S.A., September,
http://www.pcworld.com/reviews/article.asp?aid=18596.

Microsoft Corporation (2002), "Microsoft Speech API SDK", April,
http://www.microsoft.com/speech/techinfo/apioverview/.

Mohr, R. (2002), "Vocola Information Pages", May, http://home.attbi.com/~vocola.

Myers, B., Hudson, S.E., Pausch, R. (2000), "Past, Present, and Future of User Interface Software Tools", ACM Transactions on Computer-Human Interaction, pp. 3-28, Vol. 7, No. 1, March.

Nass, C., Gong, L. (2000), "Speech Interfaces from an Evolutionary Perspective", Communications of the ACM, pp. 36-43, Vol. 43, No. 9, September.

Pallett, D.S., Fourcin, A. (1995), "Speech Input: Assessment and Evaluation", Chapter 13 Section 6 in "Survey of the State of the Art in Human Language Technology", Cole et al Eds., National Science Foundation, November.

Peckam, J.B. (1984), "Automatic Speech Recognition - a solution in search of a problem?", Logic Ltd, England, Behaviour and Information Technology, Vol. 3, No. 2, pp. 145-152.

Rogerson, D.E. (1997), "Inside COM", Microsoft Press, Redmond, Washington, U.S.A.

Rosenfeld, R., Olsen, D., Rudnicky, A. (2001), "Universal Speech Interfaces", Interactions, Vol. 8, Issue 6, pp. 34-44, ACM Press, New York, NY, USA, December.

Scansoft Inc. (2002), "Dragon NaturallySpeaking Datasheets", http://www.scansoft.com/naturallyspeaking/datasheets/.

Sun Microsystems Inc. (1998a), Java Speech Grammar Format, Version 1.0, October, http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/.

Sun Microsystems Inc. (2002), Java Speech API Implementations, http://java.sun.com/products/java-media/speech/forDevelopers/jsapifaq.html#implementation.

Sun Microsystems Inc. (1998b), Java Speech API Programmers Guide, Version 1.0, October, http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide/.

Sun Microsystems Inc. (1998c), Java Speech API Specification, Version 1.0, October, http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-doc/.

Yankelovich, N. (1998), "Designing Speech User Interfaces", Tutorials, Computer Human Interaction Conference, April.

Weinschenk, S. and Barker, D.T. (2000), "Designing Effective Speech Interfaces", John Wiley & Sons, Inc., New York, NY, U.S.A.

Winig, A. and Proteau, M. (1999), "Speech User Interfaces: Basic Principles of Design", Dragon Developer Day, Dragon Systems, October.

DISTRIBUTION LIST

Interfacing COTS Speech Recognition and Synthesis Software to a Lotus Notes Military Command and Control Database

Oliver Carr

(DSTO-TR-1358)

AUSTRALIA

DEFENCE ORGANISATION

Task sponsor
Commander, DJFHQ
Head, Knowledge Systems

S&T Program
Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
Director General Science Policy Development
} shared copy
Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington (Doc Data Sheet only)
Scientific Adviser to MRDC Thailand (Doc Data Sheet only)
Scientific Adviser Joint
Navy Scientific Adviser (Doc Data Sheet and distribution list only)
Scientific Adviser - Army
Air Force Scientific Adviser
Director Trials

Information Sciences Laboratory
Chief of Command and Control Division (Doc Data Sheet and Distribution Sheet Only)
Research Leader Command & Intelligence Environments
Research Leader Theatre Operations Analysis Branch (Doc Data Sheet)
Research Leader Military Information Enterprise Branch
Head Virtual Enterprises (Doc Data Sheet)
Head Systems Simulation and Assessment (Doc Data Sheet)
Head Theatre Operations Analysis (Doc Data Sheet)
Head Information Exploitation (Doc Data Sheet)
Head Intelligence Analysis (Doc Data Sheet)
Head C2 Australian Theatre (Doc Data Sheet)
Head HQ Systems Experimentation (Doc Data Sheet)
Head Information Systems (Doc Data Sheet)
Head Human Systems Integration Group
Task Manager: Ahmad Hashemi-Sakhtsari
Author: Oliver Carr
Publications and Publicity Officer, C2D/EOC2D

DSTO Library and Archives
Library Edinburgh 2 copies
Australian Archives
Library Canberra (Doc Data Sheet)

Capability Systems Staff
Director General Maritime Development (Doc Data Sheet only)
Director General Land Development
Director General Aerospace Development (Doc Data Sheet only)

Knowledge Staff
Director General Command, Control, Communications and Computers (DGC4) (Doc Data Sheet only)

Army
SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Gallipoli Barracks, Enoggera QLD 4052
Chief of Staff, DJFHQ, Gallipoli Barracks, Enoggera, QLD 4052
Lieutenant Colonel Norm Cognet, Gallipoli Barracks, Enoggera, QLD 4052
Commander D. Abraham-James, Project Director, HQAST Projects, Electronic Systems Acquisition Division, Russell Offices, Canberra, ACT 2600
Major Mark Hughes, Information Manager, DJFHQ, Gallipoli Barracks, Enoggera, QLD 4052

Intelligence Program
DGSTA Defence Intelligence Organisation
Manager, Information Centre, Defence Intelligence Organisation

Defence Libraries
Library Manager, DLS-Canberra
Library Manager, DLS - Sydney West (Doc Data Sheet Only)

Universities and Colleges
Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Serials Section (M list), Deakin University Library
Hargrave Library, Monash University (Doc Data Sheet)
Librarian, Flinders University
Professor Robin King, Pro Vice Chancellor, Division of Information Technology, Engineering and the Environment, University of South Australia, Warrendi Road, Mawson Lakes, The Levels Campus, SA 5095
Dr Barry Dwyer, Senior Lecturer, Department of Computer Science, University of Adelaide, Adelaide, SA 5005

Other Organisations
NASA (Canberra)
State Library of South Australia
National Library of Australia

OUTSIDE AUSTRALIA

International Defence Information Centres

US Defence Technical Information Center, 2 copies
UK Defence Research Information Centre, 2 copies
Canada Defence Scientific Information Service
NZ Defence Information Centre

Abstracting and Information Organisations

Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Scientific Abstracts US
Documents Librarian, The Center for Research Libraries, US

Information Exchange Agreement Partners

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (5 copies)

Total number of copies: 56

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Interfacing COTS Speech Recognition and Synthesis Software to a Lotus Notes Military Command and Control Database

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Oliver Carr

5. CORPORATE AUTHOR

Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-1358	AR-012-484	Technical Report	October 2002

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
E-9505-23-210 (1)	JTW 01/092	COMD DJFHQ, HKS	28	40

13. URL ON THE WORLD WIDE WEB

http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-1358.pdf

14. RELEASE AUTHORITY

Chief, Command and Control Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No limitation

17. CITATION IN OTHER DOCUMENTS Yes

18. DEFTEST DESCRIPTORS

Speech recognition
Voice data processing
Systems integration
Human computer interface
Command and control systems

19. ABSTRACT

Speech recognition and synthesis technologies have become commercially viable over recent years. Two current market leading products in speech recognition technology are Dragon NaturallySpeaking and IBM ViaVoice. This report describes the development of speech user interfaces incorporating these products with Lotus Notes and Java applications. These interfaces enable data entry using speech recognition and allow warnings and instructions to be issued via speech synthesis. The development of a military vocabulary to improve user interaction is discussed. The report also describes an evaluation in terms of speed of the various speech user interfaces developed using Dragon NaturallySpeaking and IBM ViaVoice with a Lotus Notes Command and Control Support System Log database.

TECHNICAL REPORT DSTO-TR-1358 AR-012-484 OCTOBER 2002

INFORMATION SCIENCES LABORATORY
PO BOX 1500 EDINBURGH SOUTH AUSTRALIA, 5111
AUSTRALIA, TELEPHONE (08) 8259 5555